第2章

Aurora活用チュートリアル

～Xilinx社のFPGA間通信プロトコルを使いこなす

久保田新二

ここでは，FPGA間インターフェースで用いるデータ・リンク層通信プロトコルの一つである米国Xilinx社の「Aurora」の活用法を解説する．Aurora通信ブロックの生成方法やユーザ論理とのインターフェース法，検証手法について具体的に説明する．（編集部）

Auroraは，米国Xilinx社のデータ・リンク層通信プロトコルです．FPGAが搭載する高速シリアル通信ブロックRocketIOのMGTブロックを使用します（**図1**）．通信したいデータにヘッダとフッタを自動的に付加して送受信を行います（**図2**）．

SCP	Data 0	Data 1	Data 2	・・・	Data *N*-2	Data *N*-1	Data *N*	ECP

図2　シリアル・データのパケットのイメージ
通信したいデータにヘッダとフッタを自動的に付加する．

RocketIOのMGTブロックは，さまざまなシリアル通信仕様に柔軟に対応しています．しかし，特性の設定やデータ同期への考慮が難しいため，とても使用しにくいものです．これを簡単に使用できるように開発されたプロトコルが，Auroraです．

Auroraモジュールは，Xilinx社のFPGA開発ツールの一つで，IPコアの設定・生成を行う「CORE Generator」を使って作成します．ウィザードに従ってユーザ・インターフェースのパラメータと通信速度の値を設定していくだけです．難しいMGTの設定は自動で行ってくれるので，設計者はMGTをほとんど意識せずに高速シリアル通信回路を設計できます．後はAurora通信ブロックに対し，送受信したいデータをAuroraで決められたユーザ・インターフェースの仕様通りのタイミングで受け渡しすればよいのです．

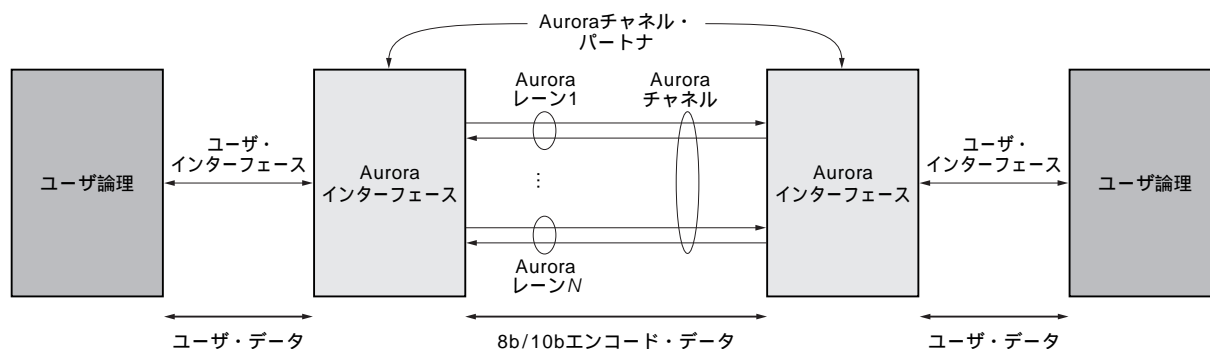



図1　Auroraを使ったシステム構成
米国Xilinx社のFPGAが搭載する高速シリアル通信ブロックRocketIOのMGTブロックを使用する．

KeyWord
FPGA，高速シリアル通信，RocketIO，Aurora，CORE Generator，MGT，クロック・コレクション，クロック，リセット，ModelSim

しかし，このインターフェースの仕様を正しく理解していないと，予想外の事態を引き起こす可能性があります．

1．高速シリアル通信ブロック活用の基礎

Auroraを使用すれば，設計者はMGTをほとんど意識せずに設計を行えると述べました．しかし，MGTの基本構成を全く知らずに設計はできません．ここではVirtex-4をターゲットに，Auroraの設計に必要な最低限の知識についてまとめます．

● MGTの構成

MGTはFPGAの品種によって搭載数が異なります．また，MGTと接続されるシリアル信号線も専用ピンとして決められており，パッケージによって異なります．

Virtex-4では，物理的にMGTは左右に分かれて配置されています（**図3**）．同様に，MGTへ供給するクロック・ドライバも左右それぞれに配置されています．この物理的要因から，供給するリファレンス・クロックは，左右それぞれにあるMGTでは共通に使用可能ですが，左右にまたがるMGTで共有することはできません．

図3　MGTの構成
Virtex-4では，物理的にMGTは左右に分かれて配置されている．MGTへ供給するクロック・ドライバも左右それぞれに配置されている．MGTは2個（MGTA，MGTB）で一つのMGTタイル（ペア）を構成する．

MGTは2個（MGTA，MGTB）で一つのMGTタイル（ペア）を構成しています．そのためMGTタイルの片側のMGTのみを使用する場合，それとペアとなっているMGTには未使用の処理が必要になります．CORE GeneratorでAuroraを作成したときに，一緒に出力されるunused_mgtというモジュールを組み込みます．さらにペアのMGT同士のコミュニケーション用にCOMBUSIN，COMBUSOUTという信号があるので，互いに接続する必要があります．

● クロック・コレクションの動作

クロック・コレクションとは，対向して接続されたAuroraの送信側から一定サイクル期間内にアイドル・シーケンスを挿入する処理のことです．

MGTの受信側には，受信データを一時蓄えるためのエラスティック・バッファというFIFOメモリが存在しています．このバッファは，書き込みと読み出しのクロックが異なります．書き込みは受信データから抽出したクロックで，読み出しはユーザ・インターフェースから供給されたクロックにて行われます．SerDesチップの場合では，受信側は従属モードとして受信したデータから抽出したクロックをユーザ・インターフェースで使用しますが，ここではクロックの乗せ換えを行っているようです．

送信側と受信側で同じ仕様の水晶発振器を使用していても，水晶発振器の精度の範囲で周波数が異なります．従って，これが原因で受信側のエラスティック・バッファにオーバフローまたはアンダフローが生じる可能性があります．すると，HARD_ERRORに至って，Lane_UPとChannel_UPがダウンし，通信が途切れてしまいます．

この問題を回避するため，一定期間内にアイドル・シーケンスを送信するサイクルが必要になります．これが送信側からのクロック・コレクションです．クロック・コレクションは，Auroraの入力ポートにあるDO_CC信号を一定期間以上アサートすることによって行います．Lane_UP，Channel_UP確立後から常に必要なサイクルです．

クロック・コレクションのアサート間隔とアサート期間は，ユーザ・インターフェースのデータ幅や水晶発振器の精度によって異なります．100ppm（ppmは百万分の1）精度では**表1**のようになります．

クロック・コレクションを挿入するためのタイミング設

図4　クロック・コレクション挿入時の送信タイミング
クロック・コレクションを挿入すると，ユーザ・データの送信ができなくなる．送信中であれば，クロック・コレクションの挿入を優先する．

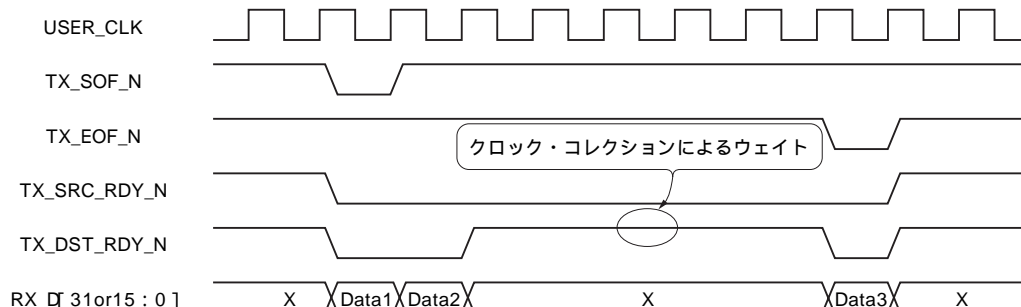



表1　クロック・コレクションのアサート間隔とアサート期間

レーン幅	DO_CCのアサート間隔（User_CLKサイクル）	DO_CCアサート期間（User_CLKサイクル）
2バイト	5000	6
4バイト	3000	3

計には注意が必要です．

Auroraモジュールでは，DO_CC信号をアサートされている間，送信側はTX_DST_RDY_N信号がデアサートされ，ユーザ・データの送信ができません（**図4**）．送信中であれば，クロック・コレクションの挿入を優先します．受信側はRX_SRC_RDY_N信号がデアサートされ，この期間は待ち状態にしなければなりません．クロック・コレクションはユーザ・データの送信よりも優先順位が高いため，データ送信中に割り込むことになります．

ユーザ・インターフェースからAuroraモジュールへ出力したデータが，実際にシリアル・データとして出力されるまでに5クロックの遅延があります．従って，フレーム・データの受信とクロック・コレクションが重複しないためには，送信終了後（EOF出力後）5クロック以上あけてからDO_CC信号をアサートする必要があります．

2. Auroraモジュールの生成

ここでは，**表2**に示すツールを使用して，Auroraモジュールを組み込む手順を具体的に説明します．

表2　Auroraモジュールの生成・検証ツール

種　類	ツール名
FPGA開発ツール	ISE 8.1 Service Pack3（WebPACKでは使用できない） CORE Generator IP Update 1（Aurora V2.4）
シミュレータ	ModelSim SE 6.1a

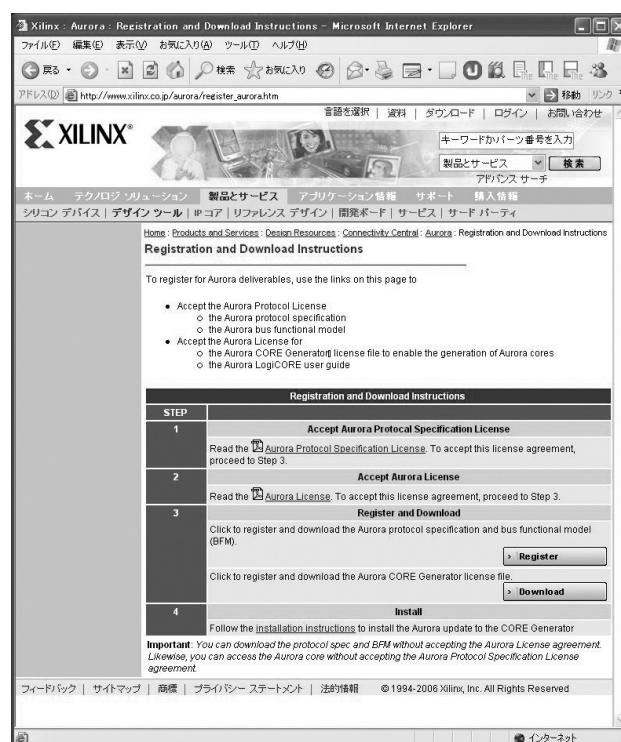



図5　Auroraコアの入手
Xilinx社のWebサイト（http://www.xilinx.co.jp/aurora/register_aurora.htm）でユーザ登録を行う．ライセンス・ファイルは電子メールで届く．

● Auroraコアの入手

Auroraを使用するためには，Xilinx社のWebサイト（ http://www.xilinx.co.jp/aurora/register_aurora.htm ）でユーザ登録を行います（**図5**）．ユーザ登録は無料です．

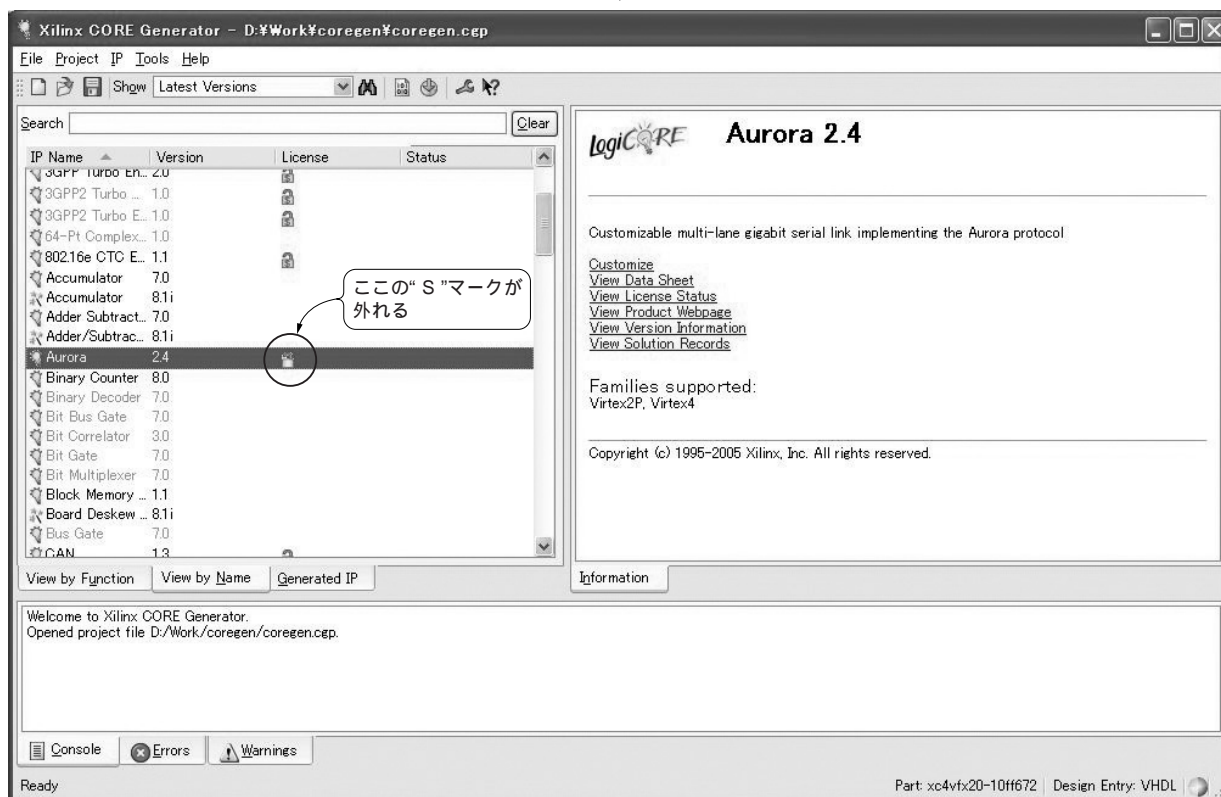
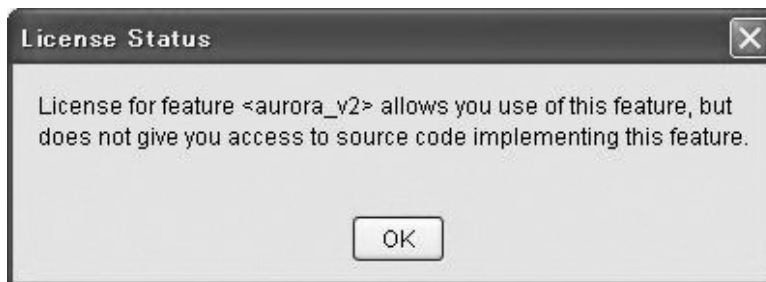
ユーザ登録が完了すると，電子メールでライセンス・ファイルが届きます．ラインセンス・ファイルは圧縮されているので，解凍した上でISEがインストールされているフォルダ先（例えば，\Xilinx\coregen\core_licenses\）にコピーするとAuroraが使用できるようになります．

次のページへ続く

図6　CORE GeneratorによるAuroraモジュールの生成
FPGA開発ツールのISEに含まれているCORE Generatorを使用する．

● CORE GeneratorによるAuroraモジュールの生成

FPGA開発ツールのISEに含まれているCORE Generatorを起動します．Auroraが正しくインストールされているとCORE GeneratorのIP Nameの横のLicense表示のマークの"S"が外され，Auroraモジュールを作成できるようになります(**図6**)．

「Aurora」の文字をダブル・クリックすると，メッセージが表示されます．ライセンスを取得済みであれば，無視してかまいません．その後，ウィザード画面が表示されるので，画面に従って各パラメータの設定を行います．

1）1ページ目の設定画面

Component Name欄に作成するAuroraモジュールのモジュール名を入力します．ここで入力したモジュール名と同じフォルダ名の下にファイルが生成されます．

Target Deviceのリストでは，使用する品種を選択します．

HDL欄では，使用する設計言語を選択します．

Aurora Lanes欄には，1チャネル内で使用するレーン数

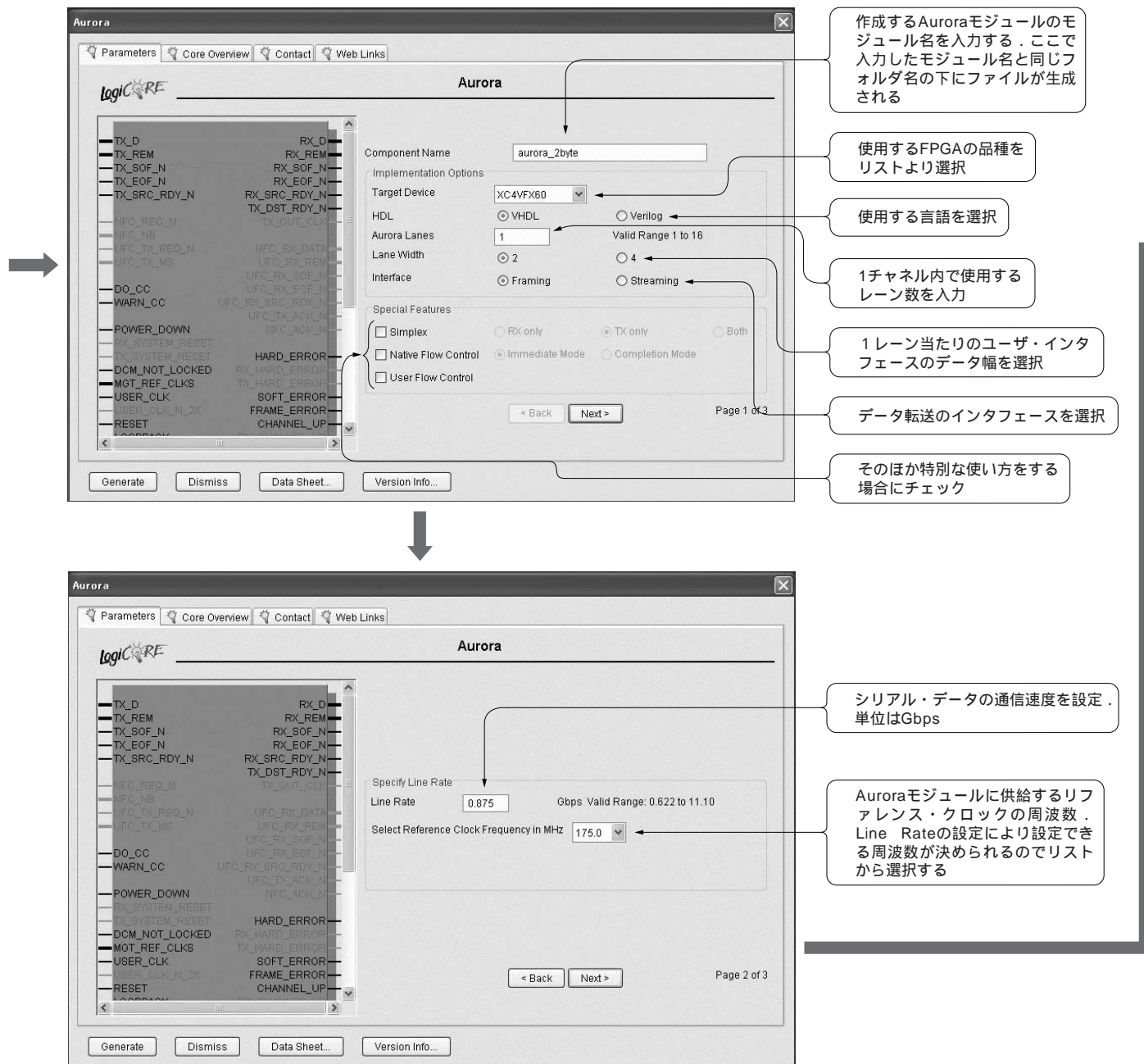



図6　CORE GeneratorによるAuroraモジュールの生成（つづき）
FPGA開発ツールのISEに含まれているCORE Generatorを使用する．

を入力します．Lane Widthでは，1レーン当たりのユーザ・インターフェースのデータ幅を選択します．単位はバイトです．Interface欄ではデータ転送のインターフェースを選択します．

Special Featuresは，別の使い方をする場合のみチェックします．

2）2ページ目の設定画面

Line Rate欄は，シリアル・データの通信速度を設定します．単位はGbpsです．Select Reference Clock Frequency in MHzは，Auroraに供給するリファレンス・クロックの周波数を選択します．Line Rateの値に応じて設定できる周波数は決められるのでリストから選択します．

3）3ページ目の設定画面

ウィザードの3ページ目では，リファレンス・クロック・ソースを選択します．Col.0とCol.1は，前に述べた（**図3**）左右にあるMGTのロケーションを示します．Col.0は左側，

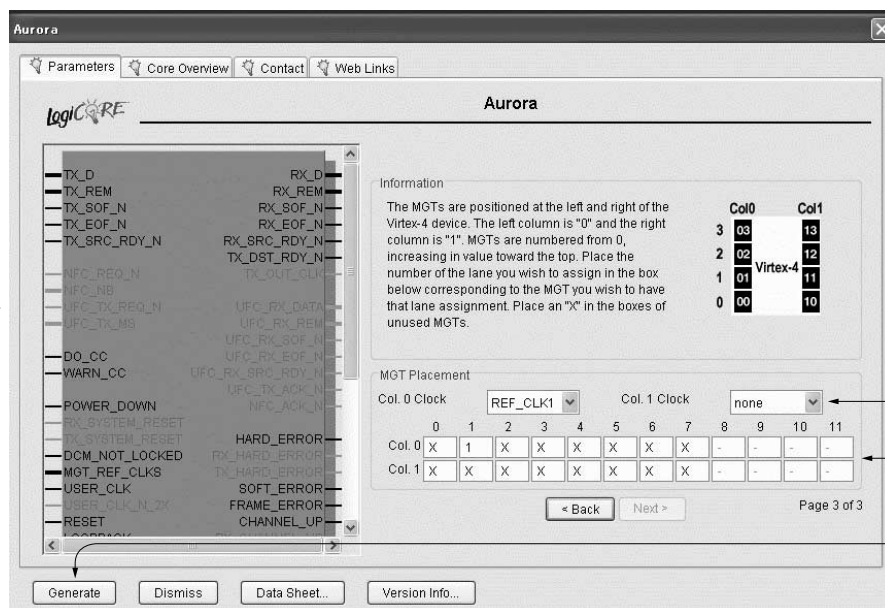



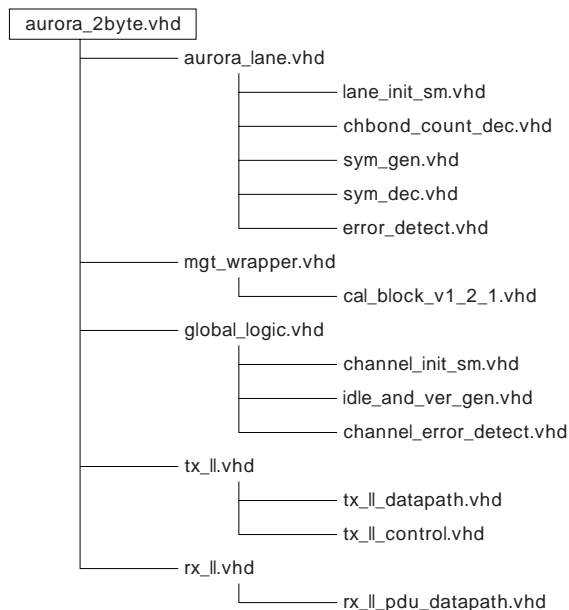
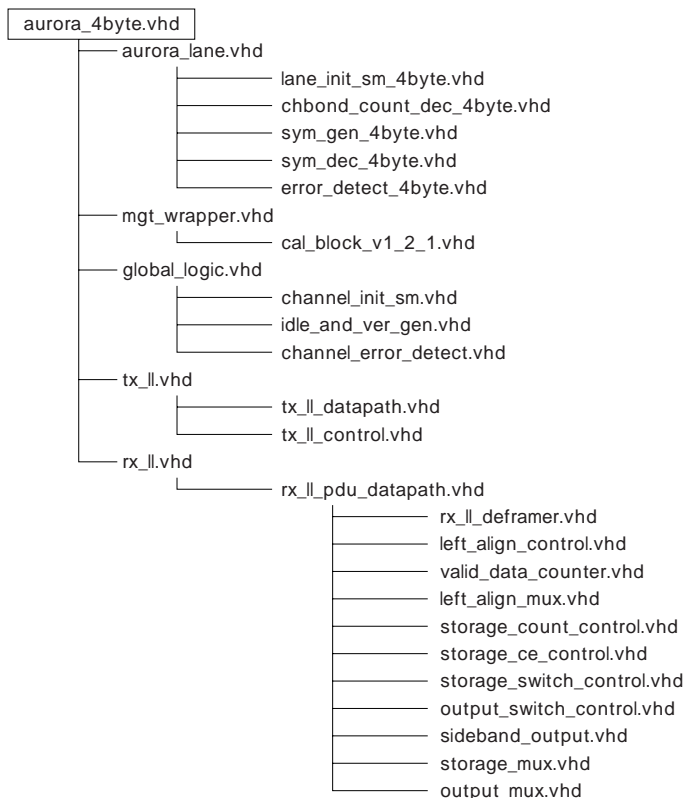
図6　CORE GeneratorによるAuroraモジュールの生成（つづき）
FPGA開発ツールのISEに含まれているCORE Generatorを使用する．

aurora_2byte
D:¥Work¥coregen¥aurora_2byte
cc_manager
clock_module
examples
scripts
src
testbench
ucf
aurora_gs_ug173.pdf
readme.txt
release_notes.txt
ug061.pdf

①クロック・コレクションの制御を行う参考回路
②DCMが組み込まれているソース・ファイル
Auroraにユーザ・クロックに同期した半分の周波数のクロックを供給するためのモジュール（2バイト・レーンのみ）
③送信側でテスト・パターンを生成し，受信側でそのパターンをチェックするサンプル回路
④サンプル回路のシミュレーション・データなど
⑤Auroraモジュールのソース・ファイル
任意で付けたモジュール名のファイルがTOPになる．下位ソース・ファイルは，そのまま使用する
⑥サンプル回路のテストベンチ
⑦制約ファイル

図7　生成されたAuroraモジュールのファイル
HDLソース・ファイルやXilinx社の評価ボード「MK421ボード（Virtex-4 RocketIO Characterization Platform）」用のサンプル回路やテストベンチなども出力される．

Col.1は右側に相当します．左右どちらのMGTを使用するかによって，リファレンス・クロックの配置もそれに合わせて設定します．さらに，その中でどのクロック線を使用するか選択します．REF_CLK1とREF_CLK2は，通信速度が1Gbps以上のときに使用します．GREF_CLKは，通信速度が1Gbpsよりも低速のときに使用します．

MGT Placementは，選択したデバイスにおいて使用するMGTの物理配置を示します．使用するMGTに，番号を1からAurora Lanesで設定した数まで入力します（未使用MGTにはXのままにする）．ここで指定した配置は，生

（**a**）2バイト・インターフェース

（**b**）4バイト・インターフェース

生成ファイル名	2バイト・インターフェース	4バイト・インターフェース	共通性
aurora_2byte.vhd	○	－	（TOP）
aurora_4byte.vhd	－	○	（TOP）
aurora_lane.vhd	○	－	－
aurora_lane_4byte.vhd	－	○	－
aurora_pkg.vhd	○	○	○
cal_block_v1_2_1.vhd	○	○	○
channel_error_detect.vhd	○	○	▲
channel_init_sm.vhd	○	○	▲
chbond_count_dec.vhd	○	－	－
chbond_count_dec_4byte.vhd	－	○	－
error_detect.vhd	○	－	－
error_detect_4byte.vhd	－	○	－
global_logic.vhd	○	○	▲
idle_and_ver_gen.vhd	○	○	▲
lane_init_sm.vhd	○	－	－
lane_init_sm_4byte.vhd	－	○	－
left_align_control.vhd	－	○	－
left_align_mux.vhd	－	○	－
mgt_wrapper.vhd	○	○	▲
output_mux.vhd	－	○	－
output_switch_control.vhd	－	○	－
rx_ll.vhd	○	○	▲
rx_ll_deframer.vhd	－	○	－
rx_ll_pdu_datapath.vhd	○	○	▲
rx_ll_ufc_datapath.vhd	○	○	▲
sideband_output.vhd	－	○	－
storage_ce_control.vhd	－	○	－
storage_count_control.vhd	－	○	－
storage_mux.vhd	－	○	－
storage_switch_control.vhd	－	○	－
sym_dec.vhd	○	－	－
sym_dec_4byte.vhd	－	○	－
sym_gen.vhd	○	－	－
sym_gen_4byte.vhd	－	○	－
tx_ll.vhd	○	○	▲
tx_ll_control.vhd	○	○	▲
tx_ll_datapath.vhd	○	○	▲
unused_mgt.vhd	○	○	▲
valid_data_counter.vhd	－	○	－

▲：生成されるファイル名は同じだが，中身が異なる

（**c**）2バイト・インターフェースと4バイト・インターフェースのファイルの共通性

図8　Auroraモジュールのソース・ファイルの構成

同じファイル名でも機能が異なるものがあるので，1チップ内に複数のモジュールを実装する場合には注意が必要．

成されるサンプルの制約ファイル(ucfファイル)に盛り込まれるだけで，後で制約ファイルを変更できるので気にする必要はありません．

最後に[Generate]ボタンをクリックすると，モジュール名と同じディレクトリにAuroraのファイルがいくつか生成されます．

● ファイル構成

CORE Generatorでは，HDLソース・ファイルが生成されます(p.67の**図7**)．また，HDLソース・ファイルのほかに，Xilinx社の評価ボード「MK421ボード(Virtex-4 RocketIO Characterization Platform)」用のサンプル回路やテストベンチなども出力されるので，初めて設計する際の参考になります．

CORE Generatorで作成されるAuroraモジュールのソース・ファイルの構成を**図8**に示します．データ幅や仕様が異なるAuroraを1チップのFPGA内に複数エントリする際，Auroraモジュールの内部で呼び出しているコンポーネントが重複してしまう問題が発生します．1チップ内に複数のAuroraモジュールを実装する場合は，同一のファイル名(コンポーネント名)で，共通性のないものはユニークな名前に変更する必要があります．

3．ユーザ回路との接続

CORE Generatorで生成されたファイルをユーザ回路と接続する方法をサンプル回路を基に説明します．

サンプル回路の構成を**図9**に示します．サンプル回路を動作させるだけであれば，クロックとリセットを供給するだけで済んでしまいます．従って，まずシリアル信号線の品質のチェックを行うには，これを組み込んでチェックするとよいでしょう．

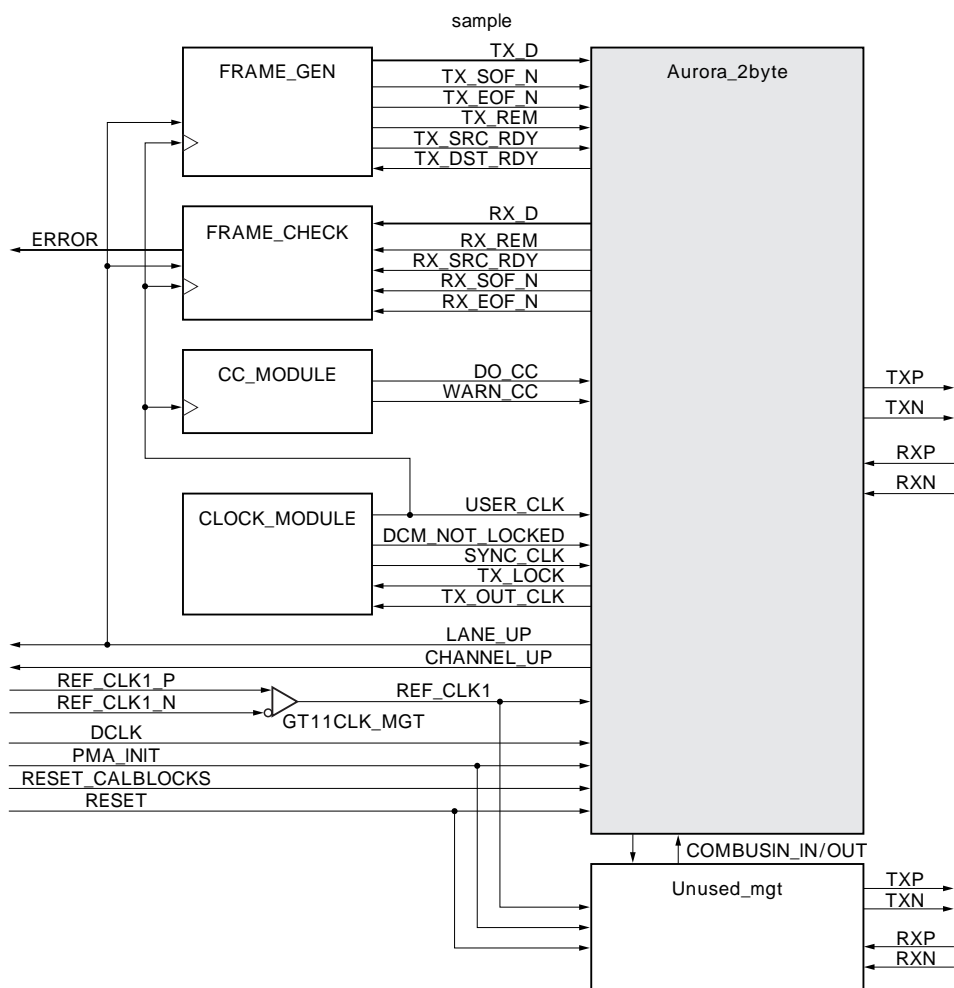



図9 Auroraモジュールを使用したサンプル回路
クロックとリセットを供給するだけで動作する．

（a）REFCLK1側の接続　　（b）REFCLK2側の接続

```
INST GT11CLK_MGT_LEFT LOC=GT11CLK_X0Y1;
INST GT11CLK_MGT_LEFT SYNCLK1OUTEN=ENABLE;
INST GT11CLK_MGT_LEFT SYNCLK2OUTEN=DISABLE;
```

（c）REFCLK1を使用する場合の設定例

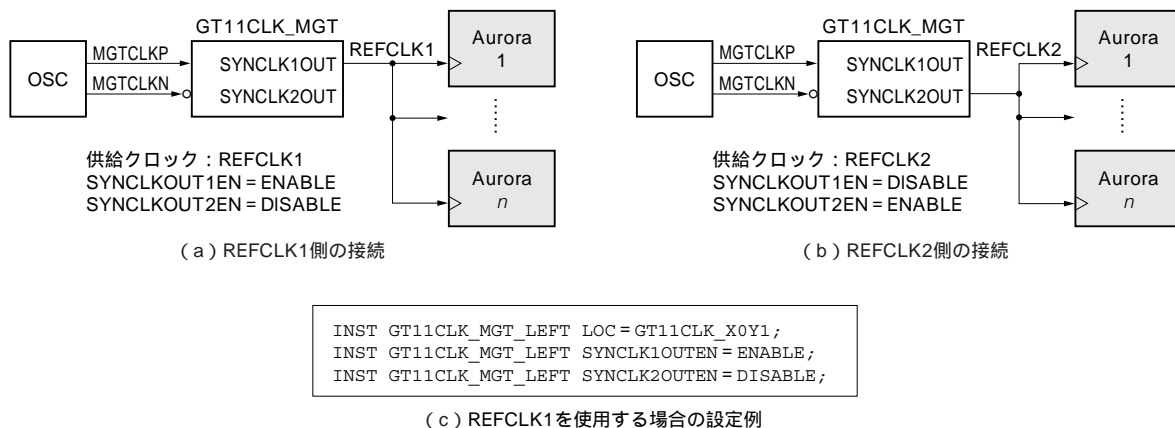



図10　リファレンス・クロックの接続
GT11CLK_MGTは使用する入力ピンに対応してチップ内の配置が決まっているので制約ファイルで設定する．

● リファレンス・クロック

Auroraへ供給するリファレンス・クロックには，3通りの接続方法があります．

- MGTCLKピンより供給（1Gbps以上で使用，ジッタ特性に優れている）
- FPGA内で生成した信号より供給（1Gbps以下で使用可）
- GREFCLKピンより供給（1Gbps以下で使用可能）

通信速度によって使用できるクロック線が異なるので注意が必要です．

サンプル回路では，水晶発振器からMGTCLKピンに供給してMGT専用のクロック・ドライバ（GT11CLK_MGT）を介して接続しています．クロック・ドライバは，Xilinx社のunisimライブラリに用意されているものです．GT11CLK_MGTからは2系統のクロックを出力できますが，CORE Generatorの生成（リファレンス・クロック・ソースの選択）において，選択したクロックと同じ信号を接続するために属性の設定をします．また，GT11CLK_MGTは使用する入力ピンに対応してチップ内の配置が決まっているので，制約ファイルで設定します（**図10**）．

● ユーザ・クロック

ユーザ論理とAuroraとの間のデータのやりとりを行うクロック（USER_CLK）は，Auroraから出力される送信クロック（TX_OUT_CLK）から生成します．TX_OUT_CLKの周波数は，送信レートとユーザ・インターフェースのデータ幅によって次のように求められます．

- 2バイト時：TX_OUT_CLK ＝（送信レート/ 40）× 2
- 4バイト時：TX_OUT_CLK ＝（送信レート/ 40）

送信レートが3.125Gbps，ユーザ・インターフェースが2バイトの場合であれば，

TX_OUT_CLK ＝（3125/40）× 2 ＝ 156.25［MHz］

となります．

ユーザ・インターフェースのデータ幅によってユーザ・クロックの生成方法が異なります．

2バイトのデータ幅の場合を**図11**に示します．USER_CLKと，このクロックに同期した半分の周波数のクロック（SYNC_CLK）の2系統のクロックが必要です．TX_OUT_CLKをCLOCK_MODULE（DCM）に入力して，TX_OUT_CLKと同じ周波数のクロック（USER_CLK）と分周したクロックを供給します．CORE Generatorで生成されたサンプル回路（clock_module）をそのまま流用できます．また同一仕様のAuroraを複数使用する場合で，配置が左右のどちらか一方だけであれば，ユーザ・クロックは共有して使用できます．Aurora回路ごとにDCMを1個ずつ使用してしまうと，DCMのリソースが足りなくなってしまうので，できるだけ共有した方がよいでしょう．

4バイトのデータ幅の場合を**図12**に示します．SYNC_CLKは必要ないため，TX_OUT_CLKをループさせてUSER_CLKに接続します．図では，BUFGを挿入していますが，特に意識して入れなくてもISEで自動的に挿入してくれます．

● DCLK

Auroraのユーザ・ガイドには掲載されていませんが，

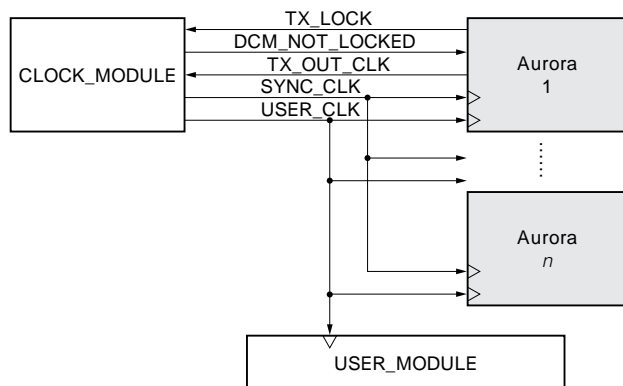



図11　2バイト・レーン時のユーザ・クロック
USER_CLKとこのクロックに同期した半分の周波数のクロック（SYNC_CLK）の2系統のクロックが必要．TX_OUT_CLKをCLOCK_MODULE（DCM）に入力してTX_OUT_CLKと同じ周波数のクロック（USER_CLK）と分周したクロックを供給する．

図12　4バイト・レーン時のユーザ・クロック
TX_OUT_CLKをループさせてUSER_CLKに接続する．

DCLKピンに25MHz～50MHzのクロックを供給しなければなりません．

リファレンス・クロックやユーザ・クロックとは別に，DRP（Dynamic Reconfiguration Port）と同じクロック線を使用して，常に動作しているクロックを供給する必要があります．これはMGTと接続しているCalibration Blockというモジュールに供給されてNBTI対策などに使用されるものです．また，このクロックの周期をAuroraのTOPモジュールのGeneric文に設定する必要があります．50MHzのクロックを接続した場合は，以下のようになります（単位はns）．

```
DCLK_PERIOD_NS_P : integer := 20;
```

● リセット信号

リセット信号は3種類あり，それぞれ以下の用途に使用します．

- PMA_INIT

MGTの内部にあるPMAと呼ばれるアナログの部分を初期化します．電源投入時からアサートしておかなければなりません．

- RESET_CALBLOCKS

MGTと接続しているCalibration Blockモジュールを初期化します．ユーザ・ガイドに載っていないので注意が必要です．

- RESET

Auroraのプロトコル関係のモジュールの初期化を行います．

リセットを解除するには，PMA_INIT→RESET_CALBLOCKSの順番で行います．RESET信号はどのタイミングでも問題ないようです．Auroraは初期化シーケンスが定義されているので，結果的にMGTが正常な受信状態になるまでステートが動かないからです．

● ユーザ・インターフェース

ユーザ論理からAurora回路を制御するためのインターフェースを，**表3**に示します．データの送受信は，タイミングに合わせて制御を行います．

● Generic文の設定変更

AuroraのTOPモジュールには，いくつかのGeneric文の設定が必要です．この中で設定が必要な項目を**表4**にまとめます．

● 未使用のMGTの扱い

MGT2個（MGTA，MGTB）で一つのMGTタイル（ペア）を構成しているので，片側のMGTのみを使用する場合，未使用側にunused_mgtモジュールを接続してMGTを保護する必要があります．未使用モジュールのソース・ファイルも，CORE Generatorで自動生成されます．ただし，どちらのMGTも使用しない場合は不要です．

インターフェースの信号は，Auroraモジュールと同じ信号名のものは，Auroraモジュールと同じように接続するだけです．

2

表3
ユーザ論理からAurora回路を制御するためのインターフェース

信号名	入出力	詳　細
TX_D	Input	送信データ・バス
TX_SOF_N	Input	送信パケットの先頭データを指定
TX_EOF_N	Input	送信パケットの最終データを指定
TX_REM	Input	送信パケットの最終データの有効バイトを指定
TX_SRC_RDY_N	Input	有効な送信データであることを指定
TX_DST_RDY_N	Output	シリアル側の送信準備ができていることを示す

（**a**）データ送信時に使用

信号名	入出力	詳　細
RX_D	Output	受信データ・バス
RX_SOF_N	Output	受信パケットの先頭データを示す
RX_EOF_N	Output	受信パケットの最終データを示す
RX_REM	Output	受信パケットの最終データの有効バイトを示す
RX_SRC_RDY_N	Output	有効な受信データであることを示す

（**b**）データ受信時に使用

信号名	入出力	詳　細
CHANNEL_UP	Output	通信が確立していることを示す
LANE_UP	Output	レーンの初期化が完了していることを示す
SOFT_ERROR	Output	受信データにエラーが発生したことを示す
FRAME_ERROR	Output	フレームやプロトコル・エラーが発生したことを示す
HARD_ERROR	Output	通信上，致命的なエラーが発生したことを示す
LOOPBACK	Input	ループバックの設定
POWER_DOWN	Input	MGTの電源断を設定

（**c**）ステータスなど

表4
TOPモジュールの変更

Generic文の記述	説　明
SIMULATION_P : integer := 0;	通常はデフォルトの‘0’のままでよい． シミュレーションするときに‘1’を設定すると初期化時のLane_UPまでのシミュレーション時間を短縮できる
LANE0_GT11_MODE_P : string := "A"; LANE0_MGT_ID_P : integer := 0	2個で一つのタイルを構成しているMGTのうち， A側を使用する場合はそれぞれ“A”，0の設定をし， B側を使用する場合は“B”，1の設定にする
DCLK_PERIOD_NS_P : integer := 20;	DCLKに供給するクロックの周波数を設定する

● データ送信のタイミング

制御信号に合わせて送信データをAuroraに入力します（**図13**）．TX_SOF_Nで先頭データを，TX_EOF_Nで最終データを指定します．TX_SRC_RDY_NとTX_DST_RDY_Nが両方ともアサートされているときに，データがシリアルから送信されます．TX_DST_RDY_Nがデアサートされているときはデータの送信を一時停止させなければなりません．

● データ受信のタイミング

Auroraから制御信号に合わせて受信データを出力してきます（**図14**）．RX_SOF_Nで先頭データを，RX_EOF_Nで最終データを指定します．RX_SRC_RDY_Nで有効な受信データであることを指定します．

NFC（Native Flow Control）またはUFC（User Flow Control）を使用しない場合に送受信においてデータが待たされる要因は，基本的にクロック・コレクションしかありません．しかし，例外があります．Auroraのユーザ・データ幅が4バイト・レーンでは，RX_SRC_RDY_N信号は受信フレームの終わりから1サイクル前に必ずデアサートされます．

これは送信するデータにSCPとECPを付加するためです．TX_DST_RDYがデアサートされるのと同様に，受信したパケットからデータなどを分離する際のレイテンシの

USER_CLK
TX_SOF_N
TX_EOF_N
TX_REM　X　REM　X
TX_SRC_RDY_N
TX_DST_RDY_N
TX_D[31or15：0]　X　Data1　X　Data2　Data3　X
一時停止

図13
データ送信のタイミング
TX_SOF_Nで先頭データを，TX_EOF_Nで最終データを指定する．TX_SRC_RDY_NとTX_DST_RDY_Nが両方ともアサートされている時にデータが，シリアルから送信される．

USER_CLK
RX_SOF_N
RX_EOF_N
RX_REM　X　REM　X
RX_SRC_RDY_N
RX_D[31or15：0]　X　Data1　Data2　X　Data3　X
4バイト・レーンでは必ずデアサートされる

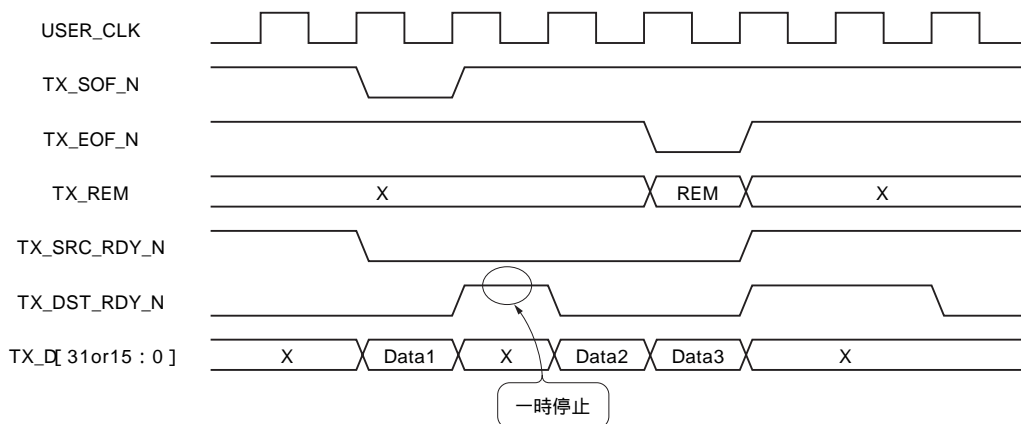



図14
データ受信のタイミング
RX_SOF_Nで先頭データを，RX_EOF_Nで最終データを指定する．RX_SRC_RDY_Nで有効な受信データであることを指定する．

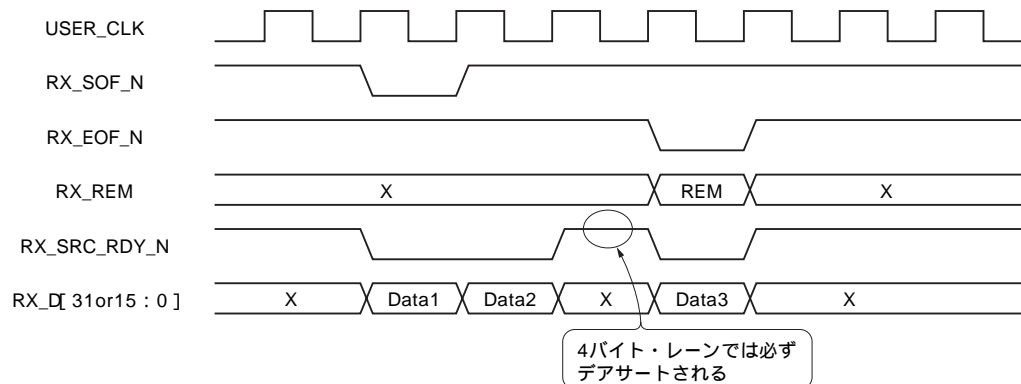
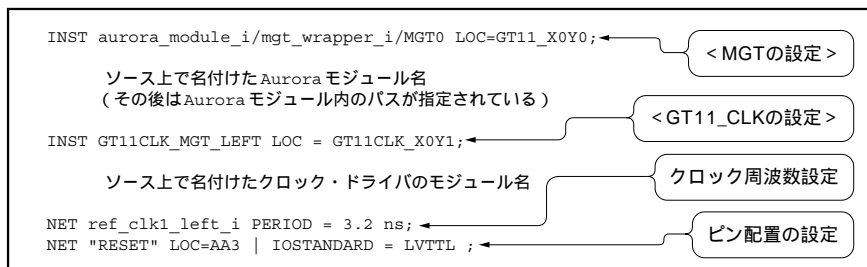



図15
制約ファイルの設定
使用するボードに合わせてucfファイルを編集する．

問題でRX_SRC_RDYがデアサートされるようです．

SCPやECPは，もともと2バイトで定義されているので，2バイト・インターフェースの受信ではEOPの前にRX_SRC_RDYをデアサートすることなくインターフェースできます．しかし，4バイトの場合だとクロック・サイクルが半分になってしまうので，このような現象になってしまうようです．

● 制約ファイルの設定

各種の制約をucfファイルに設定します．CORE Generatorで生成されたファイルを参考にしてください．ただし，aurora_sample.ucfファイルは，Xilinx社の評価ボード「MK421ボード」をターゲットにしたものなので，使用するボードに合わせてucfファイルを編集します(**図15**)．

●ロケーションの設定

MGT(GT11)とMGT専用のクロック・ドライバ(GT11_CLK)のロケーションを設定します．

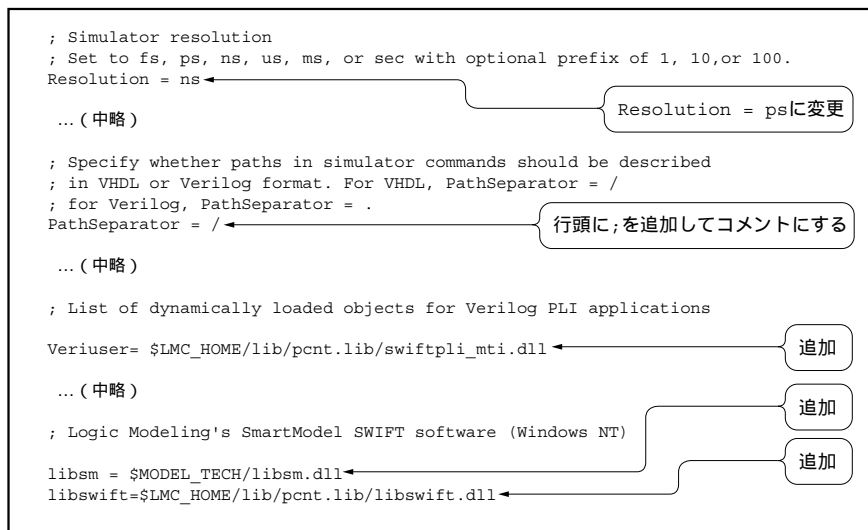
●クロック周波数設定

使用するリファレンス・クロック，ユーザ・クロック，DCLKのそれぞれの周波数の制約を設定します．

●ピン配置の設定

Virtex-4は，デフォルトの入出力ピンのレベルがLVCMOS_25です．このため，通常使用している3.3Vレベルのインターフェースでは，各ピンをLVTTLに設定する必

```
; Simulator resolution
; Set to fs, ps, ns, us, ms, or sec with optional prefix of 1, 10,or 100.
Resolution = ns   ← Resolution = psに変更

…（中略）

; Specify whether paths in simulator commands should be described
; in VHDL or Verilog format. For VHDL, PathSeparator = /
; for Verilog, PathSeparator = .
PathSeparator = /   ← 行頭に;を追加してコメントにする

…（中略）

; List of dynamically loaded objects for Verilog PLI applications

Veriuser= $LMC_HOME/lib/pcnt.lib/swiftpli_mti.dll   ← 追加

…（中略）

; Logic Modeling's SmartModel SWIFT software (Windows NT)

libsm = $MODEL_TECH/libsm.dll   ← 追加
libswift=$LMC_HOME/lib/pcnt.lib/libswift.dll   ← 追加
```

図16
ModelSimを使うためのmodelsim.iniファイルの編集
このほかに，ライブラリの追加や環境変数の設定も必要．

要があります．また，Auroraのシリアル信号線は，MGTの配置を設定していれば，ピン設定を行わなくても済むようです．設定する場合，レベルはLVDS_25にします．

● コンパイル時の注意

CORE Generatorから生成されたソース・ファイルやサンプル・ファイル一式をISEのプロジェクトに追加してコンパイルを実行します．

Generating Programming fileを実行すると，シリアル信号線に関して警告メッセージが表示されますが，配置配線（PAR）にて完全に配線が完了している場合は問題ないようです．

4．検証手法

MGTを含む設計を検証するためには，SmartModelがインストールされ，SWIFTインターフェースをサポートするシミュレータが必要です．

ここでは米国Mentor Graphics社のModelSimを使ったシミュレーションの方法を説明します．

● ModelSimによるシミュレーション

ModelSimは，SE版およびPE版（オプションで付加）がサポートされますが，この機能を使用するためにはデフォルト設定を変更する必要があります．Windows版以外は，Xilinx社のアンサーシートを参照してください．

まず，Xilinxライブラリ（unisim/simprim/XilinxCoreLib）を設定します．「File」→「New」→「Library」により，ISEがインストールされているフォルダからライブラリを追加します．

次に，modelsim.iniファイルを**図16**のように編集します．

さらに，環境変数の設定を行います．Windowsのスタート・メニューから「設定」→「コントロール・パネル」を開きます．システムの「詳細設定」タブにある「環境変数」ボタンをクリックし，ユーザ環境変数PATHに以下を追加します．

```
%LMC_HOME%¥lib¥pcnt.lib
```

また，システム環境変数「LMC_HOME」が，例えば以下のように設定されていることを確認します．

```
C:¥Xilinx¥smartmodel¥nt¥installed_nt
```

ModelSimを起動して，ModelSimのコマンド・ウィンドウで，次のコマンドを入力してエラーが表示されていなければ，シミュレータは正しく設定されています（Warningは表示される）．

```
VSIM> vsim unisim.ppc405
```

これまでの設定でシミュレータは動作しない場合があります．その場合，ModelSimのプロジェクト・ファイル（プロジェクト名.mpf）をテキスト・エディタで開いて，上記(1)～(3)の設定を直接編集してしまいましょう．それでもエラーが表示される場合は別に原因がありそうです．

Auroraのモジュールをシミュレーションするには，作成されたソース・ファイルをそのまま使用できるわけではないので注意が必要です．

Lane_UPまでのシミュレーション時間を短縮させるため

に，Auroraモジュールの`Generic`の設定値を '1 'に変更します．

リファレンス・クロックの周波数設定は，通信レートを10，20，40で割った値になるように設定しなければなりません．この設定値になっていないとLane_UPが確立できません．例えば，通信速度が3.125 Gbpsであれば，使用できるリファレンス・クロックは次の値になります．

- 312.5MHz（10分の1）
- 156.25MHz（20分の1）

そのほかの値に設定してしまうと，シミュレーションできません．

● 実機による動作試験

実機による動作試験は，まずループバック試験でシリアル信号線の品質を確認します．

シリアル信号線がケーブルへ接続されているのであれば，送信と受信をクロスで結んでループバックします．プリント基板上で接続されているのであれば，片側のAuroraはユーザ・インターフェースで受信データをそのまま送信データに接続してループバックさせるように作成します．また，単独で試験を行うのであれば，Loopback信号の設定をSerial Loopbackモードに変えて確認する方法もあります．

いずれの方法にしても，最初に確認するのはLane_UPとChannel_UPが確立されていることです．これが確立していなければ，何の通信試験も行えません．初期化状態からずっと確立していないのであれば，クロックやリセット信号が疑われます．確立と切断を繰り返しているようであれば，クロック・コレクションの挿入方法，または信号品質などの問題で通信不良が発生していることが考えられます．この判断方法としては，HARD_ERRORが発生する前に，MGTのコアの信号であるRXNOTINTABLE（8b/10bデコード・エラー）あるいはRXDISPERR（ディスパリティ・エラー）信号がアサートされているようなら，信号品質の問題と思われます．

また，テスト・パターンを使った通信試験では問題なく動作していても，実データの通信試験を開始するとHARD_ERRORが発生して，時々Lane_UPが切断されてしまうこともあります．テスト・パターンとデータの遷移状態が異なるために，このようなことも起こり得るので十分に注意が必要です．

● まとめ

Auroraは，シリアル通信のインターフェースをCORE Generatorで自動生成します．また，ユーザ・インターフェースも比較的単純です．このおかげで，FPGAの設計はそれほど難くありませんでした．しかし，工程を進めて行くうちにコンパイルでエラーが発生する，シミュレータが起動しない，実機動作ではLane_UPができないなど，次々と難題が襲ってきました．これらは，MGTの構成や使い方などを知っていれば，難なく解決できるものがありました．ここでは，それらに陥らないためのポイントについて述べてきたつもりなので，参考になることを願っています．

くぼた・しんじ
中央システム技研（株）

2

Design Wave Mook　好評発売中

動作原理，設計・製造工程から応用事例まで

MEMS開発&活用スタートアップ

Design Wave Magazine編集部 編　B5変型判 216ページ　定価2,520円（税込）ISBN4-7898-3716-5

「MEMS（micro electoro mechanical system）」や「シリコン・マイクロマシン」といったことばをよく耳にするようになりました．実際，この技術を応用したデバイスは，インク・ジェット・プリンタや自動車のエア・バッグ・システム，プロジェクタなど，さまざまな製品に搭載されています．小型，高精度，低消費電力という特徴を持つMEMSデバイスの応用分野は，今後さらに広がると期待されています．

本書は，MEMS技術の入門書であり，各種MEMSデバイスの動作原理や製造プロセス，応用について，わかりやすく説明しています．製造プロセスについては実際のMEMS工場の装置の写真を多用し，また応用については具体的な設計事例を紹介しています．

CQ出版社　〒170-8461 東京都豊島区巣鴨1-14-2 販売部 ☎（03）5395-2141 振替 00100-7-10665